C# Advanced 2021 – 2022

# Werken met Tekstbestanden

## StreamWriter

Wordt gebruikt om waardes in een bestand te schrijven.

Using(StreamWriter sw = new StreamWriter("TestBestand.Txt")) {}

zorgt er voor dat je sw.close niet meer moet doen.

// SCHRIJVEN

// Tekstbestand creëren en openen (automatisch in standaardirectory van je project \bin\Debug).

StreamWriter sw = new StreamWriter("TestBestand.Txt");

// Naar tekstbestand schrijven.

sw.WriteLine("Volgorde van getallen :"); Volgorde van getallen :

for (int i = 0; i < 10; i++) 0 1 2 3 4 5 6 7 8 9

sw.Write(i + " ");

sw.WriteLine(); // lege regel wegschrijven

// Sluiten van bestand.

sw.Close();

// SCHRIJVEN

FileStream fs = new FileStream("MyTest.Txt", FileMode.Append, FileAccess.Write);

StreamWriter sw = new StreamWriter(fs);

sw.WriteLine(" Volgorde van getallen:");

for (int i = 0; i < 10; i++)

sw.Write(i + " ");

sw.WriteLine(); // Lege regel toevoegen.

// Sluiten van bestanden.

sw.Close();

fs.Close();

## StreamReader

Wordt gebruikt om waardes te lezen uit een bestand.

Info idem sw

// LEZEN

StreamReader sr = new StreamReader("TestBestand.Txt");

// Inlezen van bestand per record.

while (! sr.EndOfStream)

{

Console.WriteLine(sr.ReadLine());

}

// Volledig bestand inlezen.

Console.WriteLine(sr.ReadToEnd());

// Inlezen van bestand per record in List

List<string> namenLijst = new List<string>();

while (!sr.EndOfStream)

{

namenLijst.Add(sr.ReadLine());

Console.WriteLine(namenLijst[j]);

j++;

}

// Sluiten van bestand.

sr.Close();

// LEZEN

FileStream fsRead = new FileStream("MyTest.Txt", FileMode.Open, FileAccess.Read);

StreamReader sr = new StreamReader(fsRead);

// Bestand tot einde lezen.

Console.Write(sr.ReadToEnd());

// Inlezen van bestand per regel.

while (!sr.EndOfStream)

{

TxtResultaat.AppendText(sr.ReadLine());

}

//Objecten sluiten

sr.Close();

fsRead.Close();

## Foutmeldingen

Text

Description automatically generated with medium confidence

## Bestanden opvragen

// Bestand opvragen

string pad = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments);

string bestand = Path.Combine(pad, "MijnTekstbestand.Txt");

// Toont alle bestandsnamen

string[] bestanden = Directory.GetFiles(TxtMap.Text);

foreach (string file in bestanden)

{

TxtBestanden.AppendText(file);

TxtBestanden.AppendText(Environment.NewLine);

}

// Toont alle mappen

string[] mappen = Directory.GetDirectories(TxtMap.Text);

foreach (string dir in mappen)

{

TxtMappen.AppendText(dir);

TxtMappen.AppendText(Environment.NewLine);

}

## OpenFileDialogue

OpenFileDialog ofd = new OpenFileDialog

## **OpenFileDialogue**

OpenFileDialog ofd = new OpenFileDialog

{

Filter = "Alle bestanden (\*.\*)|\*.\*|Tekstbestanden (\*.txt) |\*.txt",

FilterIndex = 2,

FileName = "punten.txt",

Multiselect = true,

InitialDirectory = Environment.GetFolderPath(Environment.SpecialFolder.MyDocuments)

// == OF ==

InitialDirectory = System.IO.Path.GetFullPath(@"..\..\Bestanden"), // volledig pad

// == OF ==

InitialDirectory = Environment.CurrentDirectory // onder ..\Debug

}

if (ofd.ShowDialog() == true)

{

TxtFile.Text = $"PAD & FILE: {ofd.FileName}\r\n"; //volledig pad en bestandsnaam

TxtFile.Text += $"PAD: {System.IO.Path.GetDirectoryName(ofd.FileName)}\r\n"; // enkel pad

TxtFile.Text += $"FILE: {System.IO.Path.GetFileName(ofd.FileName)}\r\n\r\n"; // enkel bestandsnaam

string pad = System.IO.Path.GetDirectoryName(ofd.FileName);

string[] bestanden = System.IO.Directory.GetFiles(pad); //Lijst van bestanden.

int i = 0;

for (; i < bestanden.Length; i++)

{

TxtFile.Text += $"{bestanden[i]}\r\n"; // pad + bestandsnaam

TxtFile.Text += $"{System.IO.Path.GetFileName(bestanden[i])}\r\n";//bestandnaam

}

TxtFile.Text += $"Aantal bestanden: {i}";

// Selectie van meerdere bestanden afdrukken in listbox.

foreach (string filename in ofd.FileNames)

{

LbxFiles.Items.Add(System.IO.Path.GetFileName(filename));

}

TxtFile.Text = $"PAD & FILE: {sfd.FileName}\r\n"; //volledig pad en bestandsnaam

TxtFile.Text += $"PAD: {System.IO.Path.GetDirectoryName(sfd.FileName)}\r\n"; // enkel pad

TxtFile.Text += $"FILE: {System.IO.Path.GetFileName(sfd.FileName)}\r\n"; // enkel bestandsnaam

TxtFile.Text += $"CURRENT DIRECTORY: {System.IO.Directory.GetCurrentDirectory()}"; // C:..\Debug

}

## **SaveFileDialogue**

SaveFileDialog sfd = new SaveFileDialog()

{

Filter = "Alle bestanden (\*.\*)|\*.\*|Tekstbestanden (\*.txt)|\*.txt",

FilterIndex = 2,

Title = "Geef een bestandsnaam op",

OverwritePrompt = true, //Bevestiging wordt gevraagd bij overschrijven van een bestand.

AddExtension = true, // Extensie wordt toegevoegd.

DefaultExt = "txt",

FileName = "Voorbeeld.txt",

//InitialDirectory = @"C:\SCHOOL\CSharp 2\Voorbeelden2", // Fysieke map

InitialDirectory = Environment.CurrentDirectory// onder ..\Debug

}

sfd.ShowDialog();

TxtFile.Text = $"PAD & FILE: {sfd.FileName}\r\n"; //volledig pad en bestandsnaam

TxtFile.Text += $"PAD: {System.IO.Path.GetDirectoryName(sfd.FileName)}\r\n"; // enkel pad

TxtFile.Text += $"FILE: {System.IO.Path.GetFileName(sfd.FileName)}\r\n"; // enkel bestandsnaam

TxtFile.Text += $"CURRENT DIRECTORY: {System.IO.Directory.GetCurrentDirectory()}"; // C:..\Debug

}

# Werken met meerdere vensters

De opstart volgorde van vensters wijzigen doe je in de App.xaml

StartupUri="HoofdvensterKlasse.xaml"

# Classes

public string Naam

{

get { return familienaam; } *geeft waarde terug*

set { familienaam = value;} *ontvangt waarde*

}

(read only)

public string Gegevens

{

get { return voornaam + " " + familienaam.ToUpper(); }

}

OF

public string Gegevens => Voornaam + " " + Naam.ToUpper();

***Method***

public string ToonNaam()

{

return voornaam + " - " + familienaam;

}

Keywords:

Partial class & 🡪 Als je klasse te groot of te ingewikkeld wordt, kan je de klasse opsplitsen in verschillende delen. Je gebruikt hierbij het sleutelwoord partial.

Static class 🡪

Abstract class 🡪 kan je geen instantie van maken, wordt altijd overgeerfd

Sealed class 🡪 Kan NOOIT overgeerfd worden

Table

Description automatically generated

CLASS LIBRARY MAKEN 🡪 ZIE PAGINA 49 IN CURSUS

# Overerving en Polymorphisme

Graphical user interface, text, application

Description automatically generated

# Linq Table Description automatically generated

Table

Description automatically generated

# WPF Besturingselementen

Verschillende pagina tabs + icons

<TabControl HorizontalAlignment="Stretch" VerticalAlignment="Stretch">

<TabItem>

<TabItem.Header>

<StackPanel Orientation="Horizontal">

<Image Source="Afbeeldingen/brief\_Case.png" />

<TextBlock Text=" Tabblad 1"/>

</StackPanel>

</TabItem.Header>

<Grid Background="#FFE5E5E5">

<TextBlock Text="Dit is het eerste tabblad. Alle controls worden binnen het Gridontworpen."/>

</Grid>

</TabItem>

<TabItem>

<TabItem.Header>

<StackPanel Orientation="Horizontal">

<Image Source="Afbeeldingen/book\_Open.png" />

<TextBlock Text=" Tabblad2"/>

</StackPanel>

</TabItem.Header>

</TabItem>

<Grid Background="#FFE5E5E5">

<TextBlock Text="Dit is het tweede tabblad. "/>

</Grid>

</TabControl>

Statusbalk

<StatusBar DockPanel.Dock="Bottom" Height="30" VerticalAlignment="Top" >

<TextBlock Text=" Ready" FontStyle="Italic" FontSize="16"/>

<Image Source="Afbeeldingen/CSharpLogo.png" Margin="10,0,0,0"/>

<Image Source="Afbeeldingen/FileFromWeb\_6281.png" Margin="5,0,0,0"/>

<TextBlock x:Name="TextBlockDatumTijd" Text="Datum en tijd" Margin="400,0,0,0"/>

</StatusBar>

Password box

<PasswordBox x:Name="PwdBoxLogin" MaxLength="4" PasswordChar="\*"/>

# ADO.NET

Zie stappenplannen connected en disconnected

**static void Main()**

**{**

**string connectionString =**

**"Data Source=(local);Initial Catalog=Northwind;"**

**+ "Integrated Security=true";**

**// Provide the query string with a parameter placeholder.**

**string queryString =**

**"SELECT ProductID, UnitPrice, ProductName from dbo.products "**

**+ "WHERE UnitPrice > @pricePoint "**

**+ "ORDER BY UnitPrice DESC;";**

**// Specify the parameter value.**

**int paramValue = 5;**

**// Create and open the connection in a using block. This**

**// ensures that all resources will be closed and disposed**

**// when the code exits.**

**using (SqlConnection connection =**

**new SqlConnection(connectionString))**

**{**

**// Create the Command and Parameter objects.**

**SqlCommand command = new SqlCommand(queryString, connection);**

**command.Parameters.AddWithValue("@pricePoint", paramValue);**

**// Open the connection in a try/catch block.**

**// Create and execute the DataReader, writing the result**

**// set to the console window.**

**try**

**{**

**connection.Open();**

**SqlDataReader reader = command.ExecuteReader();**

**while (reader.Read())**

**{**

**Console.WriteLine("\t{0}\t{1}\t{2}",**

**reader[0], reader[1], reader[2]);**

**}**

**reader.Close();**

**}**

**catch (Exception ex)**

**{**

**Console.WriteLine(ex.Message);**

**}**

**Console.ReadLine();**

**}**